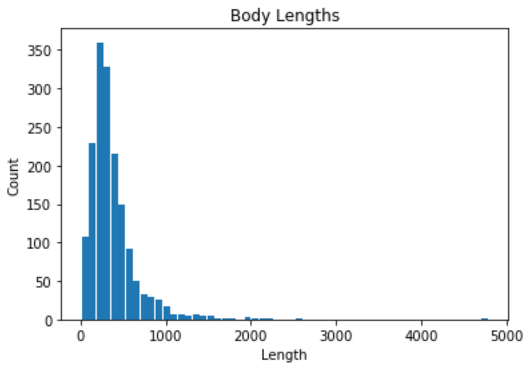
“ Grab ’em by the Fallacy ”
Stance Detection to Identify Fake News

CS585, UMass Amherst, Fall 2017

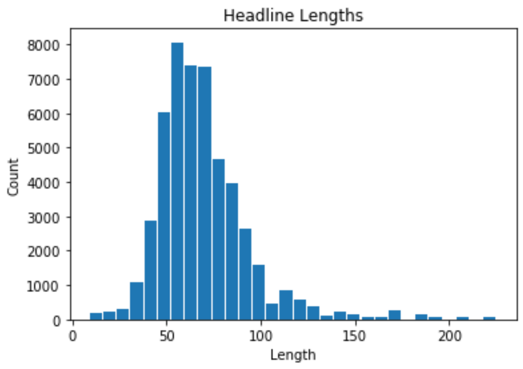
Ajinkya Zadbuke **Arhum Savera**
azadbuke@cs.umass.edu asavera@umass.edu

1 Data

The dataset we are using is the one provided for the original FNC challenge. This consists of 1683 article bodies and 49972 headlines. These headlines are matched with the articles and the corresponding pairs are labeled as one of ’unrelated’, ’agree’, ’disagree’, ’discuss’. There is a clear imbalance in the distribution of classes, with 73% of pairs being ’unrelated’, while ’disagree’ only counts for around 1.6% of the dataset. Considering this, the evaluation scheme we use is weighted (3:1) to prefer reward stance classification more than relatedness. The below figures show length of the article bodies and headlines in tokens.



(a) Dataset Body Lengths



(b) Dataset Headline Lengths

2 Features

2.1 Word overlapping features

The first feature is the Jaccard similarity between the headline and the body of the article. After tokenizing the headlines and bodies, we calculate the Jaccard similarity for each headline and body pair.

2.2 Refuting features

We use a defined list of refuting words of length n with words such as ’fake’,’fraud’,’hoax’,’false’ etc. and return a n dimensional binary vector for each headline with each dimension being 1 if the corresponding refuting word is present in the headline.

2.3 Polarity features

In this context, we define the polarity of a string to be 1 if it contains an odd number of refuting terms or 0 if it contains an even number of refuting terms. We use the same list of refuting words described above.

We also define a polarity pair to be the polarity of a headline and body of an article. We iterate over the articles and return a polarity pair for each article.

2.4 N-Gram hits

In our experiments, we use 2,3,4,5 and 6 grams. We compute the n-grams of the headline and then count how many times they appear in the body of the article. Additionally, we keep a separate count of how many common n-grams appear in the first 256 characters of the article body.

2.5 Char-grams hits

In our experiments, we use 2,4,8 and 16 character-grams. We compute the character-grams of the headline and then count how many times they appear in the body of the article. Additionally, we keep a separate count of how many common character-grams appear in the first 256 characters of the article body.

2.6 Co-occurence count

We tokenize the headline and body and count how many times a token in the headline occurs in the article body. We also keep a separate count for how many times common tokens appear in the first 256 characters of the article body.

2.7 Co-occurence count without stop words

We tokenize the headline and body, remove stop words, and count how many times a token in the headline occurs in the article body. We also keep a separate count for how many times common tokens appear in the first 256 characters of the article body.

3 Approach

We built a feature vector from the above features as an input to our classification algorithms. The baseline implementation uses a Gradient Boosting Classifier, an ensemble model that essentially combines multiple weak learners (decision trees) to minimize a specified loss function. Other models we tested were Logistic Regression, Random Forest Classifier, Gaussian Naive Bayes and Support Vector Classifier (scikit-learn). Based on multiple cross-validation results, the Gradient Boosting model performed the best. We also tried a basic Multilayer Perceptron architecture (implemented in PyTorch), with 3 hidden layers [32, 16, 8], a step-size of 0.001, optimizing the cross-entropy loss (with log-softmax for numerical stability) and a weight parameter for handling the class imbalance. This gave results almost on par with the ensemble methods, and we are optimistic that with further tweaking and using more sophisticated models (CNNs, LSTMs) or features (GLoVE), we may be able to eke out better classification performance. The below table shows accuracies of the different models over multiple folds, the dev set and the test set.

	Gradient Boosting	Random Forest	Multilayer Perceptron	Logistic Regression	SVC
fold 0	0.790	0.796	0.760	0.767	0.778
fold 1	0.793	0.797	0.769	0.769	0.780
fold 2	0.817	0.806	0.257	0.783	0.801
fold 3	0.810	0.822	0.742	0.785	0.800
fold 4	0.795	0.795	0.617	0.778	0.789
fold 5	0.764	0.779	0.424	0.748	0.759
fold 6	0.774	0.776	0.747	0.750	0.749
fold 7	0.806	0.794	0.567	0.769	0.778
fold 8	0.820	0.811	0.649	0.796	0.807
fold 9	0.787	0.786	0.786	0.758	0.767
Dev set score	79.532	78.976	77.92	76.138	77.874
Test set score	75.200	74.097	74.47	72.711	73.541

4 Timeline

Given that we have a month remaining, and we have already acquired and pre-processed our data, our only concern is on researching and implementing models. Apart from the above mentioned improvements, which we assume might take two and a half weeks at the most, we plan to allocate a few days for hyperparameter tuning and comparing performance of the various models. The last week would be reserved for final tweaks and preparing the project report and poster presentation.

5 References

[1] Fake News Challenge - https://www.fakenewschallenge.org/

[2] Augenstein, I., Rocktaschel, T., Vlachos, A., Bontcheva, K., *Stance Detection with Conditional Bidirectional Encoding*, September 2016

[3] Davis, R., Proctor, C., *Fake News, Real Consequences: Recruiting Neural Networks for the Fight Against Fake News*, January 2017

[4] Conroy, N., Rubin, V., Chen, Y. *Automatic Deception Detection: Methods for Finding Fake News*, November 2015

[5] Ferreira, W., and Vlachos, A., *Emergent : A Novel Dataset for Stance Classification*, June 2016